```
import pandas as pd
from pandas import DataFrame
from math import log
from collections import Counter
from pprint import pprint

df_tennis = pd.read_csv('/home/sahyadri/Desktop/rantest.csv')
df_tennis.keys()[0]

def entropy(probs):
    return sum([-prob * log(prob,2)for prob in probs])

def entropy_of_list(a_list):
    cnt=Counter(x for x in a_list)
    num_instances= len(a_list)*1.0
    probs=[x/num_instances for x in cnt.values()]
    return entropy(probs)
def information_gain(df,split_attribute_name,target_attribute_name):
    df_split = df.groupby(split_attribute_name)
    nobs= len(df.index)*1.0
    df_agg_cnt=df_split.agg({target_attribute_name:[entropy_of_list,lambda x:len(x)/nobs]})[target_attribute_name
    df_agg_cnt.columns = ['Entropy','PropObservations']
    new_entropy=sum(df_agg_cnt['Entropy']*df_agg_cnt['PropObservations'])
    old_entropy=entropy_of_list(df[target_attribute_name])
    return old_entropy - new_entropy

def id3(df,target_attribute_name,attribute_names,default_class=None):
    cnt=Counter(x for x in df[target_attribute_name])
    if len(cnt)==1:
        return next(iter(cnt))
    elif df.empty or(not attribute_names):
        return default_class
    else:
        default_class = max(cnt.keys())
```

```
        return next(iter(cnt))
    elif df.empty or(not attribute_names):
        return default_class
    else:
        default_class = max(cnt.keys())
        gainz=[information_gain(df,attr,target_attribute_name) for attr in attribute_names]
        index_of_max=gainz.index(max(gainz))
        best_attr=attribute_names[index_of_max]
        tree={best_attr:{}}
        remaining_attribute_names=[i for i in attribute_names if i!=best_attr]

        for attr_val,data_subset in df.groupby(best_attr):
            subtree= id3(data_subset,target_attribute_name,remaining_attribute_names,default_class)
            tree[best_attr][attr_val]=subtree
        return tree
attribute_names=list(df_tennis.columns)
attribute_names.remove('PlayTennis')
tree=id3(df_tennis,'PlayTennis',attributes_name)
print("\n\nThe resultant Decision Tree is:\n")
pprint(tree)

```

```
The resultant Decision Tree is:

{'OutLook': {'Overcast': 'Yes',
             'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes', 'Weak ': 'Yes'}},
             'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}
```

In [2]:

```
import pandas as pd
from pandas import DataFrame
from math import log
from collections import Counter
from pprint import pprint

df_tennis = pd.read_csv('/home/sahyadri/Desktop/rantest.csv')
df_tennis.keys()[0]

def entropy(probs):
    return sum([-prob * log(prob,2)for prob in probs])

def entropy_of_list(a_list):
    cnt=Counter(x for x in a_list)
    num_instances= len(a_list)*1.0
    probs=[x/num_instances for x in cnt.values()]
    return entropy(probs)
def information_gain(df,split_attribute_name,target_attribute_name):
    df_split = df.groupby(split_attribute_name)
    nobs= len(df.index)*1.0
    df_agg_cnt=df_split.agg({target_attribute_name:[entropy_of_list,lambda x:len(x)/nobs]})[target_attribute_name
    df_agg_cnt.columns = ['Entropy','PropObservations']
    new_entropy=sum(df_agg_cnt['Entropy']*df_agg_cnt['PropObservations'])
    old_entropy=entropy_of_list(df[target_attribute_name])
    return old_entropy - new_entropy

def id3(df,target_attribute_name,attribute_names,default_class=None):
    cnt=Counter(x for x in df[target_attribute_name])
    if len(cnt)==1:
        return next(iter(cnt))
    elif df.empty or(not attribute_names):
        return default_class
    else:
        default_class = max(cnt.keys())
        gainz=[information_gain(df,attr,target_attribute_name) for attr in attribute_names]
        index_of_max=gainz.index(max(gainz))
        best_attr=attribute_names[index_of_max]
        tree={best_attr:{}}
        remaining_attribute_names=[i for i in attribute_names if i!=best_attr]

        for attr_val,data_subset in df.groupby(best_attr):
            subtree= id3(data_subset,target_attribute_name,remaining_attribute_names,default_class)
            tree[best_attr][attr_val]=subtree
        return tree
attribute_names=list(df_tennis.columns)
attribute_names.remove('PlayTennis')
tree=id3(df_tennis,'PlayTennis',attributes_name)
print('\n\nThe resultant Decision Tree is:\n')
pprint(tree)
```

```
The resultant Decision Tree is:

{'OutLook': {'Overcast': 'Yes',
```

In [2]:

```
import pandas as pd
from pandas import DataFrame
from math import log
from collections import Counter
from pprint import pprint

df_tennis = pd.read_csv('/home/sahyadri/Desktop/rantest.csv')
df_tennis.keys()[0]

def entropy(probs):
    return sum([-prob * log(prob,2)for prob in probs])

def entropy_of_list(a_list):
    cnt=Counter(x for x in a_list)
    num_instances= len(a_list)*1.0
    probs=[x/num_instances for x in cnt.values()]
    return entropy(probs)
def information_gain(df,split_attribute_name,target_attribute_name):
    df_split = df.groupby(split_attribute_name)
    nobs= len(df.index)*1.0
    df_agg_cnt=df_split.agg({target_attribute_name:[entropy_of_list,lambda x:len(x)/nobs]})[target_attribute_nam
    df_agg_cnt.columns = ['Entropy','PropObservations']
    new_entropy=sum(df_agg_cnt['Entropy']*df_agg_cnt['PropObservations'])
    old_entropy=entropy_of_list(df[target_attribute_name])
    return old_entropy - new_entropy

def id3(df,target_attribute_name,attribute_names,default_class=None):
    cnt=Counter(x for x in df[target_attribute_name])
    if len(cnt)==1:
        return next(iter(cnt))
    elif df.empty or(not attribute_names):
        return default_class
    else:
        default_class = max(cnt.keys())
        gainz=[information_gain(df,attr,target_attribute_name) for attr in attribute_names]
        index_of_max=gainz.index(max(gainz))
        best_attr=attribute_names[index_of_max]
        tree={best_attr:{}}
        remaining_attribute_names=[i for i in attribute_names if i!=best_attr]

        for attr_val,data_subset in df.groupby(best_attr):
            subtree= id3(data_subset,target_attribute_name,remaining_attribute_names,default_class)
            tree[best_attr][attr_val]=subtree
        return tree
attribute_names=list(df_tennis.columns)
attribute_names.remove('PlayTennis')
tree=id3(df_tennis,'PlayTennis',attributes_name)
print("\n\nThe resultant Decision Tree is:\n")
pprint(tree)
```

```
The resultant Decision Tree is:

{'Outlook': {'Overcast': 'Yes',
             'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes', 'Weak ': 'Yes'}},
             'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}
```

In [ ]:

```
1
```

In [2]:

```
import pandas as pd
from pandas import DataFrame
from math import log
from collections import Counter
from pprint import pprint

df_tennis = pd.read_csv('/home/sahyadri/Desktop/rantest.csv')
df_tennis.keys()[0]

def entropy(probs):
    return sum([-prob * log(prob,2)for prob in probs])

def entropy_of_list(a_list):
    cnt=Counter(x for x in a_list)
    num_instances= len(a_list)*1.0
    probs=[x/num_instances for x in cnt.values()]
    return entropy(probs)
def information_gain(df,split_attribute_name,target_attribute_name):
    df_split = df.groupby(split_attribute_name)
    nobs= len(df.index)*1.0
    df_agg_cnt=df_split.agg({target_attribute_name:[entropy_of_list,lambda x:len(x)/nobs]})[target_attribute_name
    df_agg_cnt.columns = ['Entropy','PropObservations']
    new_entropy=sum(df_agg_cnt['Entropy']*df_agg_cnt['PropObservations'])
    old_entropy=entropy_of_list(df[target_attribute_name])
    return old_entropy - new_entropy

def id3(df,target_attribute_name,attribute_names,default_class=None):
    cnt=Counter(x for x in df[target_attribute_name])
    if len(cnt)==1:
        return next(iter(cnt))
    elif df.empty or(not attribute_names):
        return default_class
    else:
```

```
    else:
        default_class = max(cnt.keys())
        gainz=[information_gain(df,attr,target_attribute_name) for attr in attribute_names]
        index_of_max=gainz.index(max(gainz))
        best_attr=attribute_names[index_of_max]
        tree={best_attr:{}}
        remaining_attribute_names=[i for i in attribute_names if i!=best_attr]

        for attr_val,data_subset in df.groupby(best_attr):
            subtree= id3(data_subset,target_attribute_name,remaining_attribute_names,default_class)
            tree[best_attr][attr_val]=subtree
        return tree
attribute_names=list(df_tennis.columns)
attribute_names.remove('PlayTennis')
tree=id3(df_tennis,'PlayTennis',attributes_name)
print("\n\nThe resultant Decision Tree is:\n")
pprint(tree)

```

```
The resultant Decision Tree is:

{'OutLook': {'Overcast': 'Yes',
             'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes', 'Weak ': 'Yes'}},
             'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}
```

In [ ]:

jupyter exp4 Last Checkpoint: Last Friday at 10:16 AM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

```
in df[target_attribute_name])

(cnt))
 attribute_names):
lass

max(cnt.keys())
on_gain(df,attr,target_attribute_name) for attr in attribute_names]
nz.index(max(gainz))
ute_names[index_of_max]
{}}
ute_names=[i for i in attribute_names if i!=best_attr]

a_subset in df.groupby(best_attr):
(data_subset,target_attribute_name,remaining_attribute_names,default_class)
tr][attr_val]=subtree

tennis.columns)
PlayTennis')
Tennis',attributes_name)
 Decision Tree is:\n")
```